***Q1. Расскажите, чем, на ваш взгляд, отличается хорошее клиентское приложение от плохого с точки зрения:***

1. Хорошее клиентское приложение:
   1. Пользователь. Интуитивно-понятное взаимодействие с приложением, удобочитаемость или доступность (в том числе для пользователей с ограниченными возможностями), быстродействие, привлекательность, информативность, при просмотре с карманных устройств – отсутствие перегруза батареи.
   2. Менеджер проекта. Соответствие требованиям заказчика (дизайн, ТЗ), чётко составленная документация по проекту (приложению).
   3. Дизайнер. Мне кажется, дизайнерам не особо важно, как будет реализовано приложение =) может быть, чтобы оно было выполнено как можно точнее по макету для более чёткого понимания каких-либо недочётов с возможностью в дальнейшем их безболезненно исправить.
   4. Верстальщик. Приложение, которое он ведёт с самого начала =). Иначе – легкочитаемый, семантичный код, наличие грамотного описания приложения в части, касающейся вёрстки и её взаимодействия с этапами разработки. Чёткое разделение сфер ответственности на этапах разработки и поддержки приложения.
   5. Серверный программист. Наверное, то же, что и для верстальщика, касательно участия в разработке. + Отсутствие необходимости вмешиваться в html/css/js – части приложения. Минимальная нагрузка приложения на сервер.
2. Плохое клиентское приложение:

Которое не отвечает вышеперечисленным пунктам =)

***Q2. Опишите основные особенности разработки крупных многостраничных сайтов, функциональность которых может меняться в процессе реализации и поддержки.***

***Расскажите о своем опыте работы над подобными сайтами: какие подходы, инструменты и технологии вы применяли на практике, с какими проблемами сталкивались и как их решали.***

1. Техническое задание. ТЗ необходимо для чёткого описания требований к функционалу проекта. ТЗ должно: полностью описывать проект, содержать конкретные требования по задачам/функционалу на каждом этапе разработки, критерии выполнения работы.
2. Техническая документация. Документация нужна для полного описания реализованного функционала проекта. Это важный документ, необходимый для всех участников проекта, в т.ч. и для тех, кто пришёл на поддержку проекта после его запуска. Поэтому большую роль при составлении документации играет структурированность, простота (может быть, словарь терминов и определений) и исчерпываемость информации.
3. Полная команда/наём. Для реализации и поддержки крупного проекта удобнее всего наличие полноценной команды основных разработчиков в штате. Лучше контроль над работой, соблюдение принципов разработки внутри компании (инструментарий, стиль), оперативное решение вопросов/проблем, взаимодействие между разработчиками. Временный наём сотрудников, вероятно, удобен, если профиль компании не предусматривает полный цикл разработки, но такой способ, на мой взгляд, может осложнить работу над большим проектом.
4. Заказ/собственный проект. Я думаю, независимо от того, проект – заказной или собственный, наличие ТЗ и ТД обязательно. Если цели заказного проекта в общем ясны – прибыль и портфолио, то у собственного проекта они могут быть разные (если учитывать, что проект крупный, то это может быть собственный корпоративный портал). Думаю, что наличие собственного проекта целесообразно хотя бы потому, что на нём можно с меньшими рисками пробовать внедрение новых приёмов разработки. От разницы в целях расставляется приоритет между заказным и своим.
5. Разработка «под ключ»/субподряд. Принцип, по-моему похож с п.3. Если проектом занимается одна команда – лучше контроль, взаимодействие, поэтому «под ключ» должно быть надёжнее. Раздавать проект на субподряд, вероятно, удобно в целях экономии средств.

К сожалению, опыта участия в подобных проектах нету.. Но я считаю, что при разработке крупных сайтов должны существовать договорённости (может, даже правила) как об использовании глобальных технологий и/или инструментов (GitHub), которые влияют на взаимодействие между командами разработки, так и локальных (Sketch/Photoshop, Grunt/Gulp, Angular/Vue/React/..), которые применяются внутри команд.

***Q3. При разработке интерфейсов с использованием компонентной архитектуры часто используются термины Presentational Components и Container Components. Что означают данные термины? Зачем нужно такое разделение, какие у него есть плюсы и минусы?***

Я ещё не знаком с данной технологией, но, она уже в очереди для изучения =)

***Q4. Как устроено наследование в JS? Расскажите о своем опыте реализации JS-наследования без использования фреймворков.***

Принцип наследования в js работает с объектами. Он заключается в возможности объекта A наследовать свойства объекта B (объект B выступает в роли прототипа объекта A). У объекта-прототипа так же может быть прототип и эта иерархическая цепочка заканчивается (начинается =) ) объектом Object, чей прототип является null. У Object уже есть набор свойств (и свойств-методов), которые будут наследоваться всем создаваемым объектам. Таким образом, наследование реализуется по умолчанию каждый раз при создании объектов (массивов, функций – так как их прототипом так же является Object). Все вновь создаваемые объекты наследуют свойства их прототипа - Object. Но также можно создавать собственные прототипы. Например, когда нам необходимо использовать одни и те же свойства у разных объектов. Для этого создаётся объект-прототип, в который записываются необходимые свойства, которые потом можно «раздать» новым объектам.

В зависимости от способа создания объекта он может получить различные свойства. Создавая объект «в лоб», с помощью литералов (var a = {};), прототипом объекта «а» выступит Object. Используя конструктор (функция, устанавливающая свойства создаваемому объекту) или метод Object.create(object);, где параметром выступает существующий объект, создаваемый объект помимо базовых свойств получит и те, что заложены в конструкторе или в объекте-прототипе.

Без использования фреймворков мы реализуем js-наследование каждый раз, когда создаём новый объект, массив или функцию =)

В конкретных задачах я ещё не использовал «принудительное» наследование, но по мере изучения прописываю различные примеры реализации для лучшего понимания.

***Q5. Какие библиотеки можно использовать для написания тестов end-to-end во фронтенде? Расскажите о своем опыте тестирования веб-приложений.***

С тестированием я пока, к сожалению не знаком..

***Q6. Вам нужно реализовать форму для отправки данных на сервер, состоящую из нескольких шагов. В вашем распоряжении дизайн формы и статичная верстка, в которой не показано, как форма должна работать в динамике. Подробного описания, как должны вести себя различные поля в зависимости от действий пользователя, в требованиях к проекту нет. Ваши действия?***

Сначала обсужу этот момент с дизайнером, постараемся совместными усилиями что-то изобразить. Если не получится с дизайнером – с позволения тимлида или менеджера проекта, подберу оптимальное решение на свой взгляд, сохранив стилистику динамики других элементов сайта (если она есть).

***Q7. Расскажите, какие инструменты помогают вам экономить время в процессе написания, проверки и отладки кода.***

Для написания кода я использую Visual Studio Code. Для проверки и отладки использую «debugger» и консоль разработчика. Часто пользуюсь codepen.io для проверки каких-либо мелких или средних блоков.

***Q8. Какие ресурсы вы используете для развития в профессиональной сфере? Приведите несколько конкретных примеров (сайты, блоги и так далее).***

***Какие ещё области знаний, кроме тех, что непосредственно относятся к работе, вам интересны?***

Стараюсь использовать практически все ресурсы, о которых знаю или узнаю. Основными на данный момент являются: книга «JavaScript. Подробное руководство» Дэвида Флэнагана, учебник по js - <https://learn.javascript.ru/>, документация MDN - <https://developer.mozilla.org/ru/>, хотя достаточно часто перечитываю в англоязычной версии, потому что там оказывается гораздо понятнее =), нередко оказываюсь и на спецификации.

Из блогов и проч.: портал ХабраХабр - <https://habrahabr.ru/top/>, телеграмм канал «Библиотека верстальщика» - [@weblibrary](https://ru.telegram-store.com/catalog/channels/weblibrary/), youtube-каналы: HTML-Academy и Traversy Media, а также множество материалов (собственных и ссылочных) портала HTML-Academy.

Как область знаний, кроме разработки - очень сильно увлекаюсь историей (больше - отечественной). Как хобби и саморазвитие – музыка (меломан, но главное предпочтение – рок, и играю на гитаре, нередко с братом на студиях) и литература (отечественная классика).

***Q9. Расскажите нам немного о себе и предоставьте несколько ссылок на последние работы, выполненные вами.***

В школе учился отлично до 9 класса, потом учился хорошо. Поступил в МАИ и РЭА Плеханова, выбрал последнее. В университете работал, играл в КВН. После учёбы отслужил в армии. Потом нелёгкая занесла в РЖД, где прозябаю по сей день =(. С вёрсткой познакомился в 8 классе в Детском доме творчества на Полянке. Первый сайт был посвящён игре Commandos, который, к сожалению, не сохранился. В университете работал с контентом сайта вуза.

Общительный, весёлый, дружелюбный, неконфликтный =)

На сегодняшний день понял абсолютно точно – моя дорога в web-разработку.

Сейчас в кармане только неполных 3 учебных проекта:

<https://dimitriskot.github.io/trainigprojects/Sedona/> (Курортный городок Седона) - Базовый HTML и CSS (ванильные HTML, CSS, немного JS, валидность, семантика, резина, кроссбраузерность - IE11+, Chrome, Firefox, Opera, Safari, сделано по ТЗ Академии)  
  
<https://dimitriskot.github.io/trainigprojects/Pink/> (Фото-приложение Pink) - Продвинутый HTML и CSS (БЭМ, SASS, Gulp, node.js, немного JS, валидность, семантика, адаптив, кроссбраузерность - IE11+, Chrome, Firefox, Opera, Safari,​ ретинизация изображений, сделано по ТЗ Академии)  
  
<https://dimitriskot.github.io/trainigprojects/Keksobooking/> (проект не завершён)  
(сервис поиска аренды жилья в Токио - Кексобукинг) - Базовый Javascript (работа с DOM, обработка событий, модуляция, работа с сетью, работа со структурами данных, сделано по ТЗ Академии)

<https://codepen.io/dimitriskot/> - профиль на codepen.io, там есть несколько мелких рабочих заметок..

Я уверен, что смогу показать и доказать своё рвение к работе, самоотдачу, целеустремлённость и принести большую пользу делу.

Спасибо за уделённое внимание моему отклику, очень надеюсь на понимание и дальнейшее сотрудничество!